

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KỸ THUẬT MÁY TÍNH



HỆ ĐIỀU HÀNH (MỞ RỘNG) (CO201D)

PHẦN MỀM PHÂN TÍCH DỮ LIỆU VÀ DỰ ĐOÁN CẢM XÚC
TRÊN NỀN TẢNG MONGODB, KAFKA VÀ SPARK STREAMING

GVHD: Nguyễn Quang Hùng
SV thực hiện: Nguyễn Đắc Hoàng Phú – 2010514
Tạ Lê Đắc Lộc – 2010396
Trương Nguyễn Khôi Nguyên – 2010468

Tp. Hồ Chí Minh, Tháng 05/2022

Mục lục

1	Cơ sở lý thuyết	2
1.1	Data Platform	2
1.1.1	Điểm mạnh của Data Platform	2
1.1.2	Kiến trúc dữ liệu hiện đại: các yếu tố của Nền tảng dữ liệu	2
1.1.3	Dữ liệu hoạt động	3
1.1.4	Kết luận về Nền tảng dữ liệu	3
1.2	NoSQL Database	3
1.2.1	Định nghĩa	3
1.2.2	Tính năng của cơ sở dữ liệu NoSQL	3
1.2.3	Các loại cơ sở dữ liệu NoSQL	4
1.3	MongoDB	4
1.3.1	Định nghĩa về MongoDB	4
1.3.2	Các thuật ngữ hay sử dụng trong MongoDB	4
1.3.3	MongoDB hoạt động như thế nào	5
1.3.4	Lợi thế của MongoDB	5
1.4	Kafka	5
1.4.1	Khái niệm	5
1.4.2	Các thành phần cấu trúc của Kafka	5
1.4.3	Hoạt động của Apache Kafka	6
1.4.4	Tại sao Apache Kafka được sử dụng?	6
1.5	Spark Streaming kết hợp với Kafka xử lý luồng dữ liệu trực tiếp	6
1.5.1	Tổng quan về Apache Spark	6
1.5.2	Spark Streaming	6
1.5.3	Spark Streaming + Kafka Integration	7
1.6	Streamlit	8
2	Hướng đi của đề tài	10
2.1	Lấy dữ liệu	10
2.2	Xử lý dữ liệu	10
2.3	Hiển thị thông tin của dữ liệu	11
2.4	Lưu trữ dữ liệu trên mongoDB	11
2.5	Dự đoán cảm xúc với mỗi comment trong dữ liệu	11
2.5.1	Mô hình được lựa chọn - NaiveBayes	11
2.5.2	Attention Model - BERT FINE-TUNING	13
3	Dữ liệu Tiếng Việt	15
3.1	Đánh giá các mô hình	15
3.1.1	Data sử dụng cho các mô hình	15
3.1.2	PhoBert	15
3.1.3	SVM	15
3.1.4	LSTM	15
3.1.5	biLSTM	15
3.1.6	CNN	16
3.1.7	MLP	16
3.1.8	Kết luận về độ chính xác	16
3.2	Dự đoán dữ liệu của Shopee sử dụng MLP	17
4	Tài liệu tham khảo	18

1 Cở sở lý thuyết

1.1 Data Platform

Nền tảng dữ liệu(Data platform) là một tập hợp các công nghệ được tích hợp nhằm đáp ứng nhu cầu về end-to-end data của doanh nghiệp. Nó cho phép thu thập, lưu trữ, chuẩn bị, phân phối và quản lý dữ liệu, cũng như một lớp bảo mật cho người dùng và ứng dụng. Nền tảng dữ liệu là chìa khóa để mở khóa giá trị dữ liệu của các doanh nghiệp.

1.1.1 Điểm mạnh của Data Platform

Trong nhiều năm qua, các nhà cung cấp trong lĩnh vực CNTT đã cố gắng phát triển và đưa ra các giải pháp để giải quyết tình trạng tràn dữ liệu mà các công ty phải đối mặt từ cả bên trong và bên ngoài kinh doanh.

Hiện nay, công nghệ đám mây(Cloud) đang là tiêu chuẩn mới và các kho dữ liệu gốc đám mây hiện đang được xử lý song song với số lượng khổng lồ. Các đường ống dữ liệu(Data pipeline) có thể xử lý hàng terabyte dữ liệu. Việc lưu trữ đã trở nên ít tốn chi phí và nhanh chóng hơn, và các khung xử lý dữ liệu như Spark, Kafka có thể xử lý được một khối lượng lớn dữ liệu. Cơ sở dữ liệu NoSQL giúp tăng cường cho cơ sở dữ liệu quan hệ và Graph tăng cường cho các ngôn ngữ truyền thống như SQL trong khi các ứng dụng về AI / ML đã được phổ biến khắp nơi.

Mặc dù những mảng công nghệ riêng lẻ này đã phát triển, nhưng hầu hết các doanh nghiệp vẫn chưa thể tích hợp chúng. Kết quả là các silo dữ liệu thường không thể phát triển, chứa dữ liệu trùng lặp, thường lỗi thời, bị khóa trong các giải pháp độc quyền và không có lớp bảo mật nào.

Các Data Platform hiện đại đang cố gắng giải quyết vấn đề này. Đó là sự kết hợp của các công nghệ có thể tương tác, mở rộng và thay thế làm việc cùng nhau để cung cấp các nhu cầu tổng thể về dữ liệu của doanh nghiệp.

1.1.2 Kiến trúc dữ liệu hiện đại: các yếu tố của Nền tảng dữ liệu

- Cấp quyền cho người dùng
Người dùng cuối là trung tâm của kiến trúc nền tảng dữ liệu hiện đại. Thay vì bị giới hạn trong một tập hợp dữ liệu được phát triển trước và nguồn của chúng, người dùng có thể đưa dữ liệu của riêng họ lên nền tảng và phát triển đường dẫn riêng để nhập, làm sạch, phân tích và báo cáo về dữ liệu đó.
- Sức mạnh của Hybrid Cloud
Nền tảng dữ liệu hiện đại áp dụng những gì tốt nhất của cả thế giới thực(on-premise) và thế giới đám mây(cloud). On-premise đảm bảo thực hiện thay đổi tối thiểu đối với các ứng dụng kế thừa và Cloud đảm bảo khả năng mở rộng, xử lý, tính sẵn sàng cao, các ứng dụng pre-built và bảo mật.
- Lớp dữ liệu ảo, được chia sẻ
Cốt lõi của nền tảng dữ liệu hiện đại là lớp lưu trữ dữ liệu ảo có thể xử lý các định dạng dữ liệu và khối lượng công việc đa dạng. Lớp lưu trữ mang tính chất “trừu tượng hóa” hơn các thành phần của nền tảng khác. Ở cấp độ thấp, người dùng và ứng dụng sẽ truy cập nó bằng cách sử dụng một bộ giao thức và tiêu chuẩn chung như API REST. Từ góc độ sử dụng, dữ liệu này sẽ được liên kết và ảo hóa một cách minh bạch, cho phép người dùng chia sẻ và cộng tác trên đó.
- Scalable Data
Nhập, xác thực, làm sạch và chuẩn bị là chìa khóa cho một nền tảng dữ liệu. Một kiến trúc dữ liệu linh hoạt sử dụng những scalable pipelines có thể xử lý các tình huống khác nhau như: nhập hàng loạt từ các nguồn có sẵn bằng cách sử dụng API, pub / sub cho thông báo sự kiện không đồng bộ và xử lý luồng thời gian thực cho dữ liệu tốc độ cao.
- Mở rộng Processing logic
Kiến trúc xử lý của nền tảng dữ liệu hiện đại cho phép phát triển và sử dụng lại các ứng dụng hướng dịch vụ. Các ứng dụng này đảm nhận các chức năng trong một miền cụ thể và thường dựa trên công nghệ mã nguồn mở. Trong hầu hết các trường hợp nâng cao, nền tảng này cũng có thể cho phép phát triển các ứng dụng cho tương lai dựa trên AI và ML trong các không gian làm việc khác nhau.

- Quản trị từ đầu đến cuối
Dữ liệu được phân loại tự động và được gắn thẻ trong nền tảng dữ liệu. Siêu dữ liệu(Metadata) này cung cấp cho một danh mục dữ liệu toàn diện mà người dùng có thể tìm kiếm để khám phá các dữ liệu self-service. Mô hình quản trị cũng cho phép người dùng kiểm tra chất lượng và độ nhạy của dữ liệu. Cuối cùng, báo cáo dòng dữ liệu có thể hiển thị hành trình của phần tử dữ liệu thông qua hệ thống bất kỳ lúc nào.
- Self-Service Analytics
Lớp phân tích cho phép phát triển, phân tán và chia sẻ dashboard, báo cáo và sổ ghi chép tự phục vụ dựa trên các công nghệ linh hoạt. Các tổ chức có thể sử dụng các ứng dụng phân tích hiện có của họ bằng cách sử dụng các thư viện tích hợp khác nhau.
- Lớp bảo mật duy nhất
Cuối cùng, lớp bảo mật của kiến trúc dữ liệu hiện đại sẽ tóm tắt các cơ chế truy cập của các ứng dụng riêng lẻ. Nó có thể sử dụng Nhà cung cấp danh tính toàn doanh nghiệp (IdP) để xác thực và ủy quyền dựa trên vai trò để truy cập. Một kiến trúc dữ liệu vững chắc cũng đảm bảo dữ liệu được bảo vệ bằng việc tuân thủ các tiêu chuẩn quy định.

1.1.3 Dữ liệu hoạt động

Các loại nền tảng dữ liệu nêu trên chủ yếu giải quyết việc tổng hợp dữ liệu từ các nguồn khác nhau và sử dụng dữ liệu tổng hợp đó để trả lời các câu hỏi về phân tích kinh doanh.

Một loại nền tảng dữ liệu khác xử lý dữ liệu hoạt động với khối lượng lớn được sử dụng để phát triển các ứng dụng. Các nền tảng dữ liệu ứng dụng và “hoạt động” này ngày càng được lưu trữ trên đám mây để có thể mở rộng và dễ dàng sử dụng, có tính khả dụng cao và khả năng phục hồi tốt, cung cấp khả năng bảo mật dữ liệu mạnh mẽ khi nghỉ và truyền, đồng thời cho phép cô lập khối lượng công việc, giám sát hiệu suất và cảnh báo.

Một trong những nền tảng như vậy là MongoDB Atlas. Atlas là Cơ sở dữ liệu dưới dạng dịch vụ (DBaaS) từ MongoDB cho phép các tổ chức tạo ra các cụm MongoDB trên đám mây — mà không cần lo lắng về việc cung cấp cơ sở hạ tầng, vá lỗi, mở rộng quy mô, giám sát hiệu suất, tính khả dụng cao, bảo mật, sao lưu, khôi phục thảm họa, hoặc quản trị cơ sở dữ liệu.

MongoDB Atlas có thể làm việc liền mạch với các nền tảng dữ liệu khác để tăng cường khả năng của chúng. Ví dụ: nó có thể chạy các truy vấn liên kết nguyên bản trên các cụm AWS S3 và Atlas. Cho phép kết hợp cả dữ liệu hoạt động và dữ liệu lưu trữ đối tượng lịch sử trong cơ sở dữ liệu ảo và bộ sưu tập trên Atlas Data Lake.

1.1.4 Kết luận về Nền tảng dữ liệu

Nền tảng dữ liệu là chìa khóa để hiểu, quản lý và truy cập dữ liệu cho các tổ chức. Cuối cùng, nó phụ thuộc vào những gì bạn muốn làm với dữ liệu của mình và cách bạn muốn làm điều đó. Cho dù bạn xây dựng nền tảng dữ liệu khách hàng, nền tảng dữ liệu lớn hay sử dụng nền tảng dữ liệu hoạt động như MongoDB Atlas, nền tảng dữ liệu có thể mở ra tiềm năng và doanh thu mà dữ liệu của bạn đem lại.

1.2 NoSQL Database

1.2.1 Định nghĩa

Khi nói về thuật ngữ “cơ sở dữ liệu NoSQL”, ta thường sử dụng nó để chỉ các cơ sở dữ liệu non-relational. Một số cho rằng thuật ngữ “NoSQL” là viết tắt của “non SQL” trong khi một số khác nói rằng nó là viết tắt của “not-only SQL”. Dù bằng cách nào, hầu hết đều đồng ý rằng cơ sở dữ liệu NoSQL là cơ sở dữ liệu lưu trữ dữ liệu ở định dạng không gồm các bảng quan hệ như các cơ sở dữ liệu thông thường.

1.2.2 Tính năng của cơ sở dữ liệu NoSQL

Mỗi cơ sở dữ liệu NoSQL có các tính năng độc đáo của riêng nó. Ở cấp độ cao, nhiều cơ sở dữ liệu NoSQL có một số tính năng sau:

- Lược đồ linh hoạt

- Scaling theo chiều ngang
- Truy vấn nhanh nhờ vào mô hình dữ liệu linh hoạt
- Dễ sử dụng cho các nhà phát triển

1.2.3 Các loại cơ sở dữ liệu NoSQL

Có 4 loại cơ sở dữ liệu NoSQL chính gồm: document(tài liệu), key-value(khóa-giá trị), lưu trữ wide-column và graph(đồ thị).

- Cơ sở dữ liệu document lưu trữ dữ liệu trong tài liệu tương tự như các đối tượng JSON (JavaScript Object Notation). Mỗi tài liệu chứa các cặp fields và values. Các values thường có thể gồm nhiều loại như strings, numbers, booleans, arrays, hoặc objects.
- Cơ sở dữ liệu key-value là một loại cơ sở dữ liệu đơn giản hơn trong đó mỗi mục chứa các khóa và giá trị.
- Wide-column lưu trữ dữ liệu trong bảng, hàng và cột động.
- Cơ sở dữ liệu đồ thị lưu trữ dữ liệu trong các nút và các cạnh. Các nút thường lưu trữ thông tin về người, địa điểm và mọi thứ, trong khi các cạnh lưu trữ thông tin về mối quan hệ giữa các nút.

1.3 MongoDB

1.3.1 Định nghĩa về MongoDB

MongoDB là một cơ sở dữ liệu mã nguồn mở và là một cơ sở dữ liệu NoSQL hàng đầu, được hàng triệu người sử dụng. MongoDB được viết bằng ngôn ngữ C++. Ngoài ra, MongoDB là một cơ sở dữ liệu đa nền tảng, hoạt động trên các khái niệm Collection và Document, nó cung cấp hiệu suất cao, tính khả dụng cao và có khả năng mở rộng dễ dàng.

1.3.2 Các thuật ngữ hay sử dụng trong MongoDB

_id – Là trường bắt buộc có trong mỗi document. Trường _id đại diện cho một giá trị duy nhất trong document MongoDB. Trường _id cũng có thể được hiểu là khóa chính trong document. Nếu bạn thêm mới một document thì MongoDB sẽ tự động sinh ra một _id đại diện cho document đó và là duy nhất trong cơ sở dữ liệu MongoDB.

Collection – Là nhóm của nhiều document trong MongoDB. Collection có thể được hiểu là một bảng tương ứng trong cơ sở dữ liệu RDBMS (Relational Database Management System). Collection nằm trong một cơ sở dữ liệu duy nhất. Các collection không phải định nghĩa các cột, các hàng hay kiểu dữ liệu trước.

Cursor – Đây là một con trỏ đến tập kết quả của một truy vấn. Máy khách có thể lặp qua một con trỏ để lấy kết quả.

Database – Nơi chứa các Collection, giống với cơ sở dữ liệu RDMS chúng chứa các bảng. Mỗi Database có một tập tin riêng lưu trữ trên bộ nhớ vật lý. Một máy chủ MongoDB có thể chứa nhiều Database.

Document – Một bản ghi thuộc một Collection thì được gọi là một Document. Các Document lần lượt bao gồm các trường tên và giá trị.

Field – Là một cặp name – value trong một document. Một document có thể có không hoặc nhiều trường. Các trường giống các cột ở cơ sở dữ liệu quan hệ.

JSON – Viết tắt của JavaScript Object Notation. Con người có thể đọc được ở định dạng văn bản đơn giản thể hiện cho các dữ liệu có cấu trúc. Hiện tại JSON đang hỗ trợ rất nhiều ngôn ngữ lập trình.

Index – Là những cấu trúc dữ liệu đặc biệt, dùng để chứa một phần nhỏ của các tập dữ liệu một cách dễ dàng để quét. Chỉ số lưu trữ giá trị của một fields cụ thể hoặc thiết lập các fields, sắp xếp theo giá trị của các fields này. Index hỗ trợ độ phân tích một cách hiệu quả các truy vấn. Nếu không có chỉ mục, MongoDB sẽ phải quét tất cả các documents của collection để chọn ra những document phù hợp với câu truy vấn. Quá trình quét này là không hiệu quả và yêu cầu MongoDB để xử lý một khối lượng lớn dữ liệu.

Hãy lưu ý sự khác biệt của các trường và _id trong một document. Một _id được dùng để đại diện cho một document và chúng được sinh ra khi thêm một Document vào Collection.

1.3.3 MongoDB hoạt động như thế nào

- MongoDB hoạt động dưới một tiến trình ngầm service, luôn mở một cổng (Cổng mặc định là 27017) để lắng nghe các yêu cầu truy vấn, thao tác từ các ứng dụng gửi vào sau đó mới tiến hành xử lý.
- Mỗi một bản ghi của MongoDB được tự động gắn thêm một field có tên “_id” thuộc kiểu dữ liệu ObjectId mà nó quy định để xác định tính duy nhất của bản ghi này so với bản ghi khác, cũng như phục vụ các thao tác tìm kiếm và truy vấn thông tin về sau. Trường dữ liệu “_id” luôn được tự động đánh index (chỉ mục) để tốc độ truy vấn thông tin đạt hiệu suất cao nhất.
- Mỗi khi có một truy vấn dữ liệu, bản ghi được cache (ghi đệm) lên bộ nhớ Ram, để phục vụ lượt truy vấn sau diễn ra nhanh hơn mà không cần phải đọc từ ổ cứng.
- Khi có yêu cầu thêm/sửa/xóa bản ghi, để đảm bảo hiệu suất của ứng dụng mặc định MongoDB sẽ chưa cập nhật xuống ổ cứng ngay, mà sau 60 giây MongoDB mới thực hiện ghi toàn bộ dữ liệu thay đổi từ RAM xuống ổ cứng.

1.3.4 Lợi thế của MongoDB

- Ít schema hơn: Vì schema được sinh ra là để nhóm các đối tượng vào 1 cụm, để quản lý. Ví dụ như tạo 1 schema tên là Students chẳng hạn thì chỉ có những gì liên quan đến student thì mới được cho vào schema này. Trong khi đó trong mongodb thì chỉ 1 collection ta có thể chứa nhiều document khác nhau. Với mỗi document thì số trường, nội dung, kích thước lại có thể khác nhau.
- Cấu trúc của một đối tượng rõ ràng.
- Khả năng mở rộng cực lớn: việc mở rộng dữ liệu mà không phải lo đến các vấn đề như khóa ngoại, khóa chính, kiểm tra ràng buộc, ... MongoDB cho phép thực hiện replication và sharding nên việc mở rộng cũng thuận lợi hơn.
- Sử dụng bộ nhớ trong để lưu giữ của sổ làm việc cho phép truy cập dữ liệu nhanh hơn. Việc cập nhật được thực hiện nhanh gọn nhờ update tại chỗ (in-place).

1.4 Kafka

1.4.1 Khái niệm

Kafka là nền tảng streaming phân tán, có thể mở rộng và là sản phẩm mã nguồn mở. Dự án Kafka ban đầu được phát triển bởi Linkedin sau đó trở thành dự án Apache mã nguồn mở vào năm 2011. Kafka được viết bằng ngôn ngữ Scala và Java. Nó được viết ra nhằm mục đích cung cấp một nền tảng mà có độ trễ thấp và thông lượng cao cho việc xử lý các nguồn cấp dữ liệu theo thời gian thực.

Kafka là gì? – Có thể hiểu là một hệ thống logging để lưu lại các trạng thái của hệ thống để phòng tránh mất thông tin.

1.4.2 Các thành phần cấu trúc của Kafka

- PRODUCER: Kafka lưu, phân loại message theo topic, sử dụng producer để publish message vào các topic. Dữ liệu được gửi đến partition của topic lưu trữ trên Broker. CONSUMER: Kafka sử dụng consumer để subscribe vào topic, các consumer được định danh bằng các group name. Nhiều consumer có thể cùng đọc một topic.
- TOPIC: Dữ liệu truyền trong Kafka theo topic, khi cần truyền dữ liệu cho các ứng dụng khác nhau thì sẽ tạo ra các topic khác nhau.
- PARTITION: Đây là nơi dữ liệu cho một topic được lưu trữ. Một topic có thể có một hay nhiều partition. Trên mỗi partition thì dữ liệu lưu trữ cố định và được gán cho một ID gọi là offset. Trong một Kafka cluster thì một partition có thể replicate (sao chép) ra nhiều bản. Trong đó có một bản leader chịu trách nhiệm đọc ghi dữ liệu và các bản còn lại gọi là follower. Khi bản leader bị lỗi thì sẽ có một bản follower lên làm leader thay thế. Nếu muốn dùng nhiều consumer đọc song song dữ liệu của một topic thì topic đó cần phải có nhiều partition.

- **BROKER:** Kafka cluster là một set các server, mỗi một set này được gọi là 1 broker
- **ZOOKEEPER:** được dùng để quản lý và bố trí các broker.

1.4.3 Hoạt động của Apache Kafka

Kafka được xây dựng dựa trên mô hình publish/subscribe, tương tự như bất kỳ hệ thống messaging hiện đại nào khác.

Những producers gửi message tới node kafka cluster(gồm các broker) và thông tin sẽ được hiển thị bởi các ứng dụng được gọi là consumers.

Khi những messages này gửi tới kafka node thì chúng đều được lưu trữ tại nơi gọi là topic. Sau đó, consumer hoàn toàn có thể subscribe đến topic và lắng nghe các messages. Khi đó, messages có thể là thông tin bất kỳ như: văn bản, giá trị cảm biến, hành động của người dùng, ... Còn topic sẽ được xem là tên danh mục mà những message được lưu trữ trong đấy

Đa phần topics trong Kafka có kích cỡ rất lớn. Vì vậy, không nên lưu trữ toàn bộ dữ liệu của topic vào một node. Nguồn dữ liệu này nên phân chia rõ ràng thành nhiều Partitions sẽ cho phép thực hiện subscribe song song bằng biện pháp phân chia dữ liệu có trong một topic cụ thể. Mỗi một partition đều sẽ được đặt trên một máy riêng biệt và cho phép nhiều consumer có thể đọc dữ liệu từ một topic diễn ra song song.

Ngoài ra, để gia tăng sự khả dụng của partition thì mỗi partition đều sở hữu giá trị replicas của riêng nó.

1.4.4 Tại sao Apache Kafka được sử dụng?

Kafka là dự án mã nguồn mở, đã được đóng gói hoàn chỉnh, khả năng chịu lỗi cao và là hệ thống messaging nhanh. Vì tính đáng tin cậy của nó, kafka đang dần được thay thế cho hệ thống messaging truyền thống. Đây là một hệ quả khi mà khả năng mở rộng theo chiều ngang và chuyển giao dữ liệu đáng tin cậy là những yêu cầu quan trọng nhất. Một vài trường hợp sử dụng kafka:

- **Website Activity Monitoring:** theo dõi hoạt động của website
- **Stream Processing:** xử lý stream
- **Log Aggregation:** tổng hợp log
- **Metrics Collection:** thu thập dữ liệu

1.5 Spark Streaming kết hợp với Kafka xử lý luồng dữ liệu trực tiếp

1.5.1 Tổng quan về Apache Spark

Apache Spark là một framework mã nguồn mở tính toán cụm, được phát triển sơ khởi vào năm 2009 bởi AMPLab. Sau này, Spark đã được trao cho Apache Software Foundation vào năm 2013 và được phát triển cho đến nay.

Tốc độ xử lý của Spark có được do việc tính toán được thực hiện cùng lúc trên nhiều máy khác nhau. Đồng thời việc tính toán được thực hiện ở bộ nhớ trong (in-memories) hay thực hiện hoàn toàn trên RAM.

Spark cho phép xử lý dữ liệu theo thời gian thực, vừa nhận dữ liệu từ các nguồn khác nhau đồng thời thực hiện ngay việc xử lý trên dữ liệu vừa nhận được (Spark Streaming).

Spark không có hệ thống file của riêng mình, nó sử dụng hệ thống file khác như: HDFS, Cassandra, S3,... Spark hỗ trợ nhiều kiểu định dạng file khác nhau (text, csv, json...) đồng thời nó hoàn toàn không phụ thuộc vào bất cứ một hệ thống file nào.

1.5.2 Spark Streaming

Apache Spark có 5 thành phần chính là Spark Core, Spark Streaming, Spark SQL, MLlib và GraphX. Trong phạm vi nghiên cứu lần này, nhóm chỉ tập trung vào Spark Streaming cho việc phân tích dữ liệu trực tuyến trên các nền tảng mạng xã hội, trang mua bán trực tuyến.

• Khái niệm

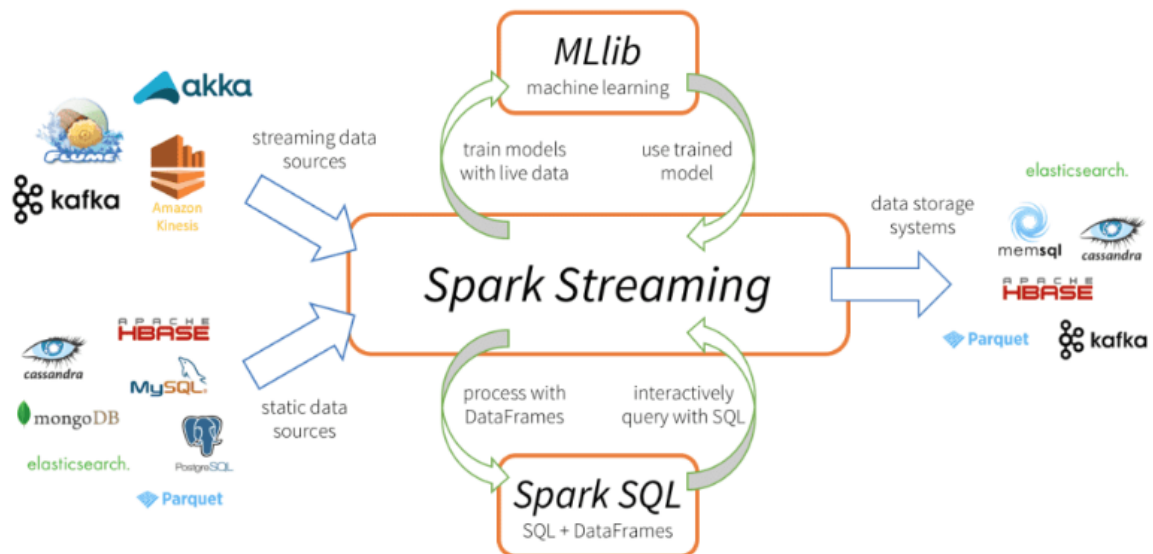
Apache Spark Streaming là một hệ thống xử lý phát trực tuyến có khả năng chịu lỗi có thể mở rộng, hỗ trợ nguyên bản cả khối lượng công việc hàng loạt và phát trực tuyến. Spark Streaming là một phần mở rộng của API Spark cốt lõi cho phép các kỹ sư dữ liệu và nhà khoa học dữ liệu xử lý dữ liệu theo thời gian thực từ nhiều nguồn khác nhau bao gồm (nhưng không giới hạn ở) Kafka, Flume và Amazon Kinesis. Dữ liệu đã xử lý này có thể được đẩy ra hệ thống tệp, cơ sở dữ liệu và trang tổng quan trực tiếp.

Tóm tắt chính của nó là Discretized Stream hay ngắn gọn là DStream, đại diện cho một dòng dữ liệu được chia thành các lô nhỏ. DStream được xây dựng dựa trên RDD, sự trừu tượng hóa dữ liệu cốt lõi của Spark. Điều này cho phép Spark Streaming tích hợp liền mạch với bất kỳ thành phần Spark nào khác như MLlib và Spark SQL.

Spark Streaming khác với các hệ thống khác có công cụ xử lý được thiết kế chỉ để phát trực tuyến hoặc có các API hàng loạt và phát trực tuyến tương tự nhưng được biên dịch nội bộ sang các công cụ khác nhau. Công cụ thực thi đơn lẻ của Spark và mô hình lập trình hợp nhất cho hàng loạt và phát trực tuyến dẫn đến một số lợi ích độc đáo so với các hệ thống phát trực tuyến truyền thống khác.

• Những lợi ích chính của Spark Streaming

- Phục hồi nhanh chóng từ các sự cố và lỗi;
- Cân bằng tải và sử dụng tài nguyên tốt hơn;
- Kết hợp dữ liệu truyền trực tuyến với tập dữ liệu tĩnh và các truy vấn tương tác;
- Tích hợp gốc với các thư viện xử lý nâng cao (SQL, học máy, xử lý đồ thị).



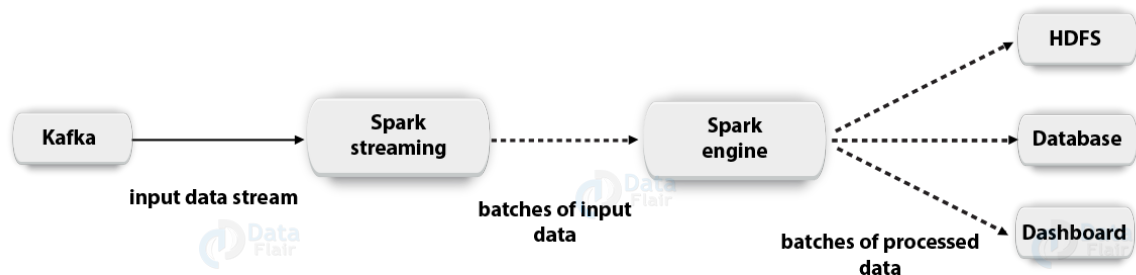
1.5.3 Spark Streaming + Kafka Integration

• Khái niệm

Trong Apache Kafka Spark Streaming Integration, có hai cách để định cấu hình Spark Streaming để nhận dữ liệu từ Kafka, tức là Kafka Spark Streaming Integration.

Đầu tiên là bằng cách sử dụng Bộ nhận và API cấp cao của Kafka, và cách thứ hai, cũng như một cách tiếp cận mới, là không sử dụng Bộ nhận. Có các mô hình lập trình khác nhau cho cả hai cách tiếp cận, chẳng hạn như các đặc tính hiệu suất và đảm bảo ngữ nghĩa.

Kafka-Spark Streaming Integration



• Cách thức hoạt động

Trong Apache Kafka Spark Streaming Integration, có hai cách để định cấu hình Spark Streaming để nhận dữ liệu từ Kafka, tức là Kafka Spark Streaming Integration.

Đầu tiên là bằng cách sử dụng Bộ nhận và API cấp cao của Kafka, và cách thứ hai, cũng như một cách tiếp cận mới, là không sử dụng Bộ nhận. Có các mô hình lập trình khác nhau cho cả hai cách tiếp cận, chẳng hạn như các đặc tính hiệu suất và đảm bảo ngữ nghĩa.

a. Phương pháp tiếp cận dựa trên bộ nhận

Ở đây, chúng tôi sử dụng Bộ nhận để thu thập dữ liệu. Vì vậy, bằng cách sử dụng API người dùng cấp cao của Kafka, ta sẽ triển khai Bộ nhận. Hơn nữa, dữ liệu nhận được được lưu trữ trong các trình thực thi Spark. Sau đó, các công việc do Kafka - Spark Streaming đưa ra sẽ xử lý dữ liệu.

Mặc dù, có khả năng cách tiếp cận này có thể làm mất dữ liệu do lỗi trong cấu hình mặc định. Do đó, cần phải bật thêm nhật ký ghi trước trong Kafka Spark Streaming, để đảm bảo không mất dữ liệu. Khi đó, tất cả dữ liệu Kafka đã nhận vào nhật ký ghi trước sẽ được lưu trên hệ thống tệp phân tán một cách đồng bộ. Bằng cách này, có thể khôi phục tất cả dữ liệu khi bị lỗi.

b. Phương pháp tiếp cận trực tiếp (Không sử dụng bộ nhận)

Sau Phương pháp tiếp cận dựa trên bộ nhận, phương pháp tiếp cận "trực tiếp" không cần bộ nhận mới đã được giới thiệu. Nó đảm bảo các điều kiện đầu cuối mạnh mẽ hơn nhiều. Cách tiếp cận này truy vấn Kafka một cách định kỳ về các offset mới nhất trong mỗi topic và các partition, thay vì sử dụng bộ nhận để thu thập dữ liệu.

Ngoài ra, xác định phạm vi offset để xử lý trong mỗi batch tương ứng. Hơn nữa, để đọc phạm vi offset được xác định từ Kafka, API người dùng đơn giản được sử dụng, đặc biệt khi các công việc để xử lý dữ liệu được khởi chạy. Tuy nhiên, việc đọc tệp từ hệ thống tệp cũng tương tự như vậy.

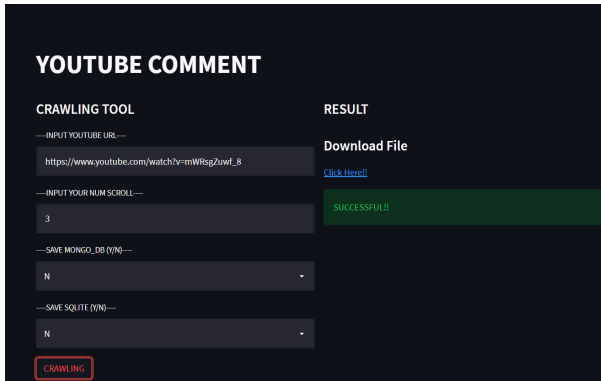
Lưu ý: Tính năng chỉ được giới thiệu trong Spark 1.3 cho API Scala và Java, trong Spark 1.4 cho API Python.

1.6 Streamlit

- Streamlit là một thư viện hỗ trợ cho data scientist dựng UI cho một ứng dụng liên quan data một cách nhanh chóng.
- Mặc dù hầu hết các web app được xây dựng bởi Flask đều có thể transfer sang streamlit, framework này vẫn chưa thật sự hoàn thiện để thay thế được Flask. Bởi vì: Streamlit vẫn còn khá mới, và chưa thật sự đảm bảo được các vấn đề về bảo mật và an toàn cho hệ thống. Streamlit chưa hỗ trợ

việc customize mạnh mẽ được như Flask (hoặc Django), hiện tại, nếu bạn muốn custom hoặc chỉnh sửa một method hiển thị theo ý bạn, bạn cần clone lại source của streamlit và chỉnh sửa trực tiếp trên source. Streamlit chưa hỗ trợ việc chuyển giao dữ liệu qua lại giữa các trang. Hiểu đơn giản là ví dụ bạn muốn truy cập vào trang xyz thì bạn cần vào trang login của xyz trước. Nhưng streamlit lại không đảm bảo được liên kết này.

- Chỉ nên sử dụng Streamlit cho vài mục đích sau: Cho dự án nhỏ sử dụng cho demo, có một dashboard duy nhất, chấp nhận layout và không yêu cầu customize theo ý của mình.



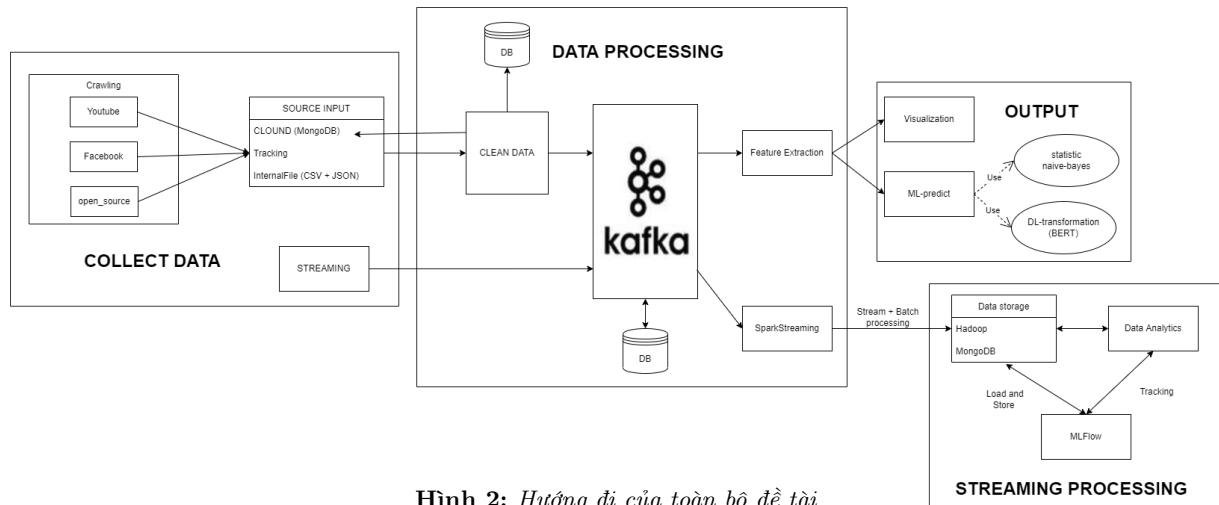
(a) UI cho việc crawl data



	user_name	time_post	comment_post	num_like	num_
0	WrathKnows	8 months ago	This song will never die!	23K	362
1	huracan2006	18 hours ago	This song never gets old. ...	19	2
2	GalaxyBoy FBI	1 day ago	The Song That Never Dies...	7	0
3	TheCheezer14	1 month ago	"When you're happy, you ...	1.1K	29
4	Shihab	20 hours ago	When the days are cold A...	5	0
5	Vancy Jarquin Diaz	3 days ago	Díos cuanto me encantan...	6	0

(b) Kết quả trả về từ UI

2 Hướng đi của đề tài



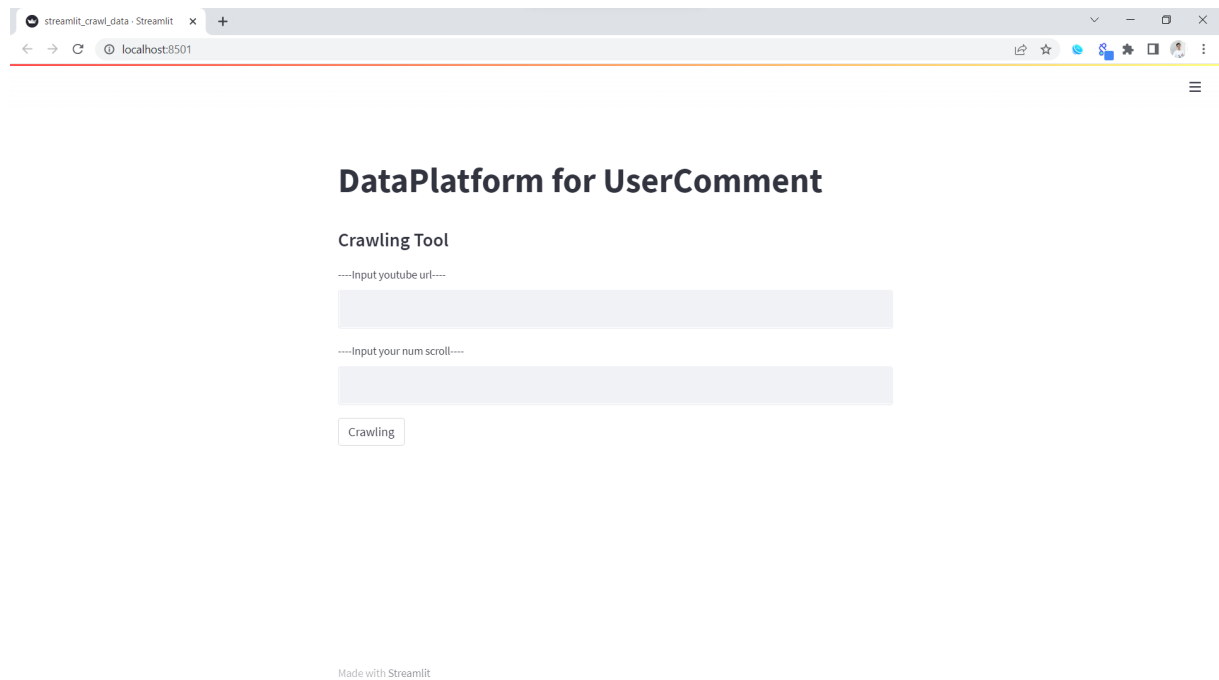
Hình 2: Hướng đi của toàn bộ đề tài

2.1 Lấy dữ liệu

Với nguồn dữ liệu là bình luận từ nền tảng Youtube. Chúng em sử dụng python và streamlit để có thể lấy bình luận từ video thông qua html. Một bình luận sẽ có các dữ liệu cần lưu là *tên người bình luận, thời gian đã đăng tải bình luận ấy, nội dung bình luận, số lượt like, số lượt bình luận để trả lời bình luận đã có*.

Để có thể lấy dữ liệu bình luận từ một video trên youtube thì bọn em cần hai thông số đó là đường link dẫn đến video và số lần cuộn trang xuống để lấy bình luận.

Mô phỏng trang để dẫn link và cài đặt về số lần cuộn trang.



2.2 Xử lí dữ liệu

Trong bước này thì chúng em sẽ làm sạch dữ liệu qua các bước như sau:

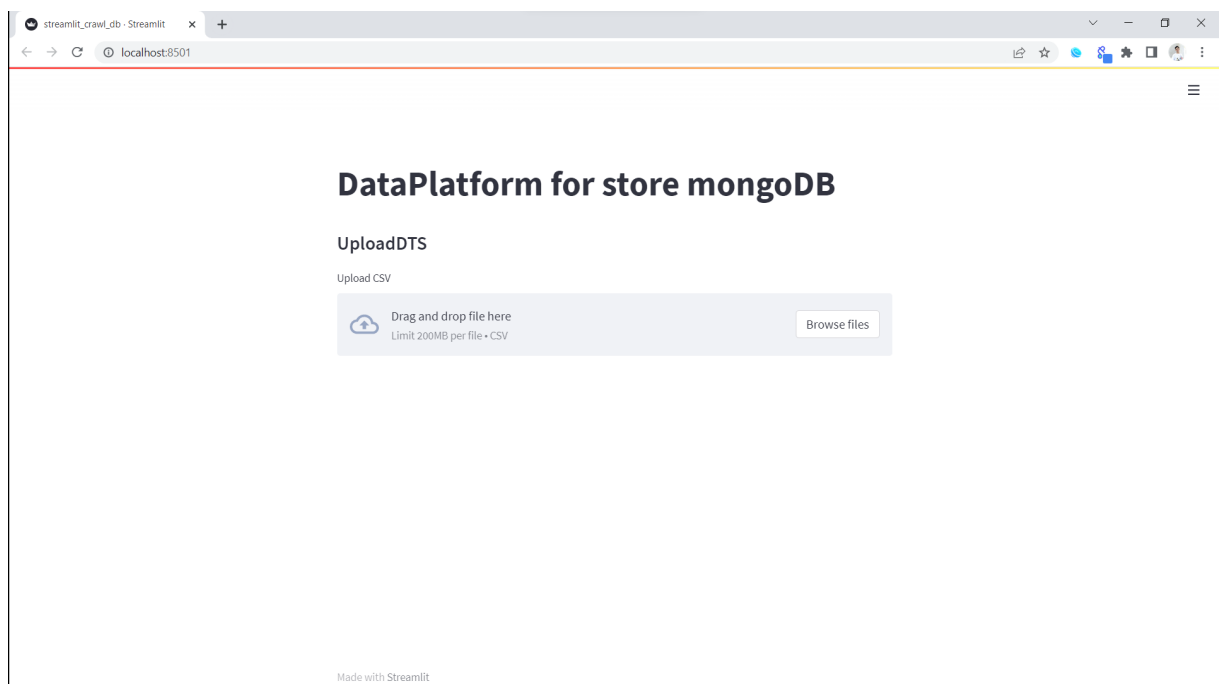
- Lọc những bình luận tiếng anh.
 - Sử dụng từ điển (tên từ điển?) có sẵn để có thể nhận ra những từ tiếng anh trong câu.
 - Đếm số từ tiếng anh đã nhận dạng được rồi đem so sánh với tổng độ dài của câu. Nếu phần trăm chữ là tiếng anh lớn hơn 80% thì giữ lại, còn không thì xoá đi dữ liệu đó đi.
- Lọc bỏ icon trong câu.
- Chuyển thì của câu về một thì thống nhất.
- Chuyển các dấu câu thành thành dấu chấm (có thể phát triển thêm để nhận diện dấu chấm thang là câu mang cảm xúc mạnh).
- Chuyển dữ liệu ở các trường khác như là trường thời gian, số lượt tương tác, số bình luận con về một dạng thống nhất để có thể thuận tiện cho việc sắp xếp.
 - Số lượt tương tác và số bình luận con: ví dụ chuyển từ 1.3k thành 1300.
 - Thời gian đã đăng bình luận: chuyển từ các đơn vị như là month hay year sang day.

2.3 Hiện thị thông tin của dữ liệu

Vẽ biểu đồ hoặc thống kê dựa trên tập dữ liệu đã được xử lí: Biểu đồ thống kê về các vấn đề như là: độ dài câu, tần suất xuất hiện của các từ và cụm từ.

2.4 Lưu trữ dữ liệu trên mongoDB

Sử dụng pymongo (với core là mongoDB) và kết nối với mongoDB Atlas để thực hiện hóa việc lưu trữ dữ liệu trên cloud.



2.5 Dự đoán cảm xúc với mỗi comment trong dữ liệu

2.5.1 Mô hình được lựa chọn - NaiveBayes

+ Naive Bayes là một thuật toán phân loại cho các vấn đề phân loại nhị phân (hai lớp) và đa lớp với các giả định yếu tố đầu vào được cho là độc lập với nhau được ký hiệu như sau: với $X = (x_1, x_2, x_3, \dots, x_n)$

thì $P(X|c) = P(x_1, x_2, x_3, \dots, x_n|c) = P(x_1|c) * P(x_2|c) * P(x_3|c) * \dots * P(x_n|c)$, tức là lúc này các x_i độc lập với nhau. Từ đó thuật toán tính xác suất cho các yếu tố đó, sau đó chọn kết quả với xác suất cao nhất.

+ Tuy nhiên dựa vào giả thuyết này mà bước training và testing trở nên vô cùng nhanh chóng và đơn giản. Chúng ta có thể sử dụng nó cho các bài toán large-scale. Trên thực tế, NBC hoạt động khá hiệu quả trong nhiều bài toán thực tế, đặc biệt là trong các bài toán phân loại văn bản, ví dụ như lọc tin nhắn rác hay lọc email spam.

+ Trong bài toán phân loại cảm xúc này, mình sẽ xây dựng trước một tập từ điển, sau đó dựa vào tập training data, với mỗi từ kí hiệu là word, ta sẽ đếm số lần nó xuất hiện trong cả câu mang nghĩa tích cực rồi tăng value(word,1) lên 1 và câu mang cảm xúc tiêu cực value(word,0) rồi tăng lên 1. Tới bước evaluate, với mỗi word, ta sẽ tính value(word,1) / value(word,0) rồi nhân hết lại với nhau, nếu tích này \geq hơn 1 ta sẽ kết luận câu này mang nghĩa tích cực, còn < 1 ta sẽ kết luận câu này mang nghĩa tiêu cực. Bên cạnh đó ta còn nhân cho giá trị đại diện cho độ cân bằng của dữ liệu trong tập training. Lưu ý thay vì nhân ta có thể lấy log rồi đưa phép nhân về phép cộng, lúc này, công việc tính toán sẽ nhẹ nhàng hơn, sẽ ít bị sai sót hơn.

Các bước xây dựng Model

```
1 def count_tweets(tweets, ys)
2     result = {}
3     for y, tweet in zip(ys, tweets):
4         lst_tweet = tweet.strip().split()
5         for word in lst_tweet:
6             tup = (word, y)
7             if tup in result.keys():
8                 result[tup] += 1
9             else:
10                result[tup] = 1
11     return result
12 #return value(word,1) and value(word,0) with every word in training_data
13
14 def train_naive_bayes(freqs, train_x, train_y)
15     ....
16     for word in vocab:
17         freqs_pos = freqs.get((word, 1), 0)
18         neg_pos = freqs.get((word, 0), 0)
19
20         #smoothing
21         freqs_pos_prob = (freqs_pos + 1) / (N_pos + V) #N_pos is number pos sentence in data_training
22         freqs_neg_prob = (neg_pos + 1) / (N_neg + V) #N_neg is number neg sentence in data_training
23
24         log_likelihood[word] = np.log(freqs_pos_prob) - np.log(freqs_neg_prob)
25
26     return log_prior, log_likelihood
27 #return log_prior (balance value of data) and log_like_li_hood (sentiment value for every word)
28
29 def naive_bayes_predict(tweet, log_prior, log_likelihood):
30     tweet = tweet.strip().split() #convert word to list
31
32     res = 0
33
34     for word in tweet:
35         if word in log_likelihood:
36             res += log_likelihood[word]
37     res += log_prior
38
39     return res
40 #evaluate value sentiment for sentence, if value < 0 -> negative sentence and vice versa
41
```

Evaluation

```
1 #function for evaluate accuracy
2 def test_naive_bayes(test_x, test_y, log_prior, log_likelihood):
3     y_hat = []
4
```

```
5     for tweet in test_x:
6         res = naive_bayes_predict(tweet, log_prior, log_likelihood)
7         if res >= 0:
8             y_hat.append(1)
9         else:
10            y_hat.append(0)
11
12    y_hat = np.array(y_hat)
13    accuracy = np.sum(test_y == y_hat)/len(y_hat)
14    return accuracy
15
```

Với tập data imdb, ta có lần lượt có các thông số evaluate sau:

+ train_data: 0.9033333333333333

+ test_data: 0.8464444444444444

+ Một số kết quả dự đoán với vài câu mẫu:

```
1     sentence = "hey you are terrible and i don't want you"
2     -> negative
3
4     sentence = "I have a bad day"
5     -> negative
6
7     sentence = "happy birthday, you are my everything"
8     -> positive
9
10    sentence = "I don't like you but you are very special, i think it is good attributes"
11    -> positive
12
```

Thông số này khá tốt và tính toán cũng khá nhanh, train chỉ mất tầm 15s, nhanh hơn nhiều so với phương pháp sử dụng deep learning. Nhưng dù thế phương pháp này vẫn có nhược điểm, đó là tính thứ tự của các từ và cả tính liên kết của từ, chẳng hạn từ: "không thích", từ này mang nghĩa tiêu cực nhưng trong naive bayes nó sẽ tách làm hai từ, "không" và "thích" để phân tích → dẫn đến dễ có nhiều sai sót. Do nó không hiểu thứ tự liên kết của câu, các từ trước có thể ảnh hưởng đến các từ sau.

Hướng phát triển:

- + Có thể thêm xác suất một từ phụ thuộc vào 2,3 từ trước hoặc 2,3 từ sau nó.
- + Lựa tập data mang tính cân bằng hơn, để tránh trường hợp log_prior nghiêng hẳn về 1 bên.
- + Thử nhiều output hơn, với nhiều mức cảm xúc chẳng hạn: positive, negative, neutral, irrelevant để xem phương pháp này có làm tốt hơn không.

2.5.2 Attention Model - BERT FINE-TUNING

* Sử dụng BERT cho bài toán:

- BERT là viết tắt của Bidirectional Encoder Representations from Transformers được hiểu là một mô hình học sâu hay còn gọi là pre-train model, học ra các vector đại diện theo ngữ cảnh 2 chiều của từ, được sử dụng để transfer sang các bài toán khác trong lĩnh vực xử lý ngôn ngữ tự nhiên. BERT đã thành công trong việc cải thiện những công việc gần đây trong việc tìm ra đại diện của từ trong không gian số (không gian mà máy tính có thể hiểu được) thông qua ngữ cảnh của nó.
- Cốt lõi của BERT chính là attention model, model sẽ bao gồm 2 phần chính là stack của encoder, stack của decoder cũng như một bảng gồm Q(Query), K(Key), V(Value) để lựa chọn từ phù hợp nhất trong ngữ cảnh đã cho.
- Sử dụng BERT cho bài toán, chúng ta phải tinh chỉnh lại model, và input đầu vào, tách ra 3 lớp: mask, type và word và lấy Bert như một layer, rồi qua lớp dropout hạn chế overfitting, để cuối cùng trả về đầu ra.

* Kiến trúc Model

```
1 def BERT_create_model():
2     input_word_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32,
3                                             name="input_word_ids")
4     input_mask = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32,
5                                         name="input_mask")
6     input_type_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32,
7                                             name="input_type_ids")
8
9     pooled_output, sequence_output = bert_layer([input_word_ids, input_mask, input_type_ids])
10
11     drop = tf.keras.layers.Dropout(0.4)(pooled_output)
12     output = tf.keras.layers.Dense(1, activation="sigmoid", name="output")(drop)
13
14     model = tf.keras.Model(
15         inputs={
16             'input_word_ids': input_word_ids,
17             'input_mask': input_mask,
18             'input_type_ids': input_type_ids
19         },
20         outputs=output)
21     return model
22
```

* Evaluation

- + Độ chính xác tập train là 98% và cả train và validation khoảng 90% dù chỉ train có 5 epoch, mỗi epoch train với batch_size = 1150, chỉ là một phần nhỏ của tập, và input đầu vào mỗi text chỉ lấy length khoảng 250.
- + Điều đó cho thấy model này thật sự còn tốt hơn nữa nếu chúng ta train với nhiều epoch hơn cũng như xử lý summary data, thì tự tin model có thể đạt tới 95% trong tập test.
- + Một số output của BERT, khắc phục được trường hợp sai của CNN và LSTM.

```
1     sentence = "hey you are terrible and i don't want you"
2     -> negative
3
4     sentence = "happy birthday, you are my everything"
5     -> positive
6
7     sentence = "I'm so terrible"
8     -> negative
9
```

* Hướng phát triển

- + Train lâu hơn với epoch, summary lại data.
- + Thử tìm hiểu thêm các pretrained model như ELMO, OpenAI so sánh với BERT.
- + Sẽ sử dụng model này cho phân tích cảm xúc với tập được crawl từ youtube, facebook.

3 Dữ liệu Tiếng Việt

3.1 Đánh giá các mô hình

3.1.1 Data sử dụng cho các mô hình

Data được chọn gồm 30000 dữ liệu crawl có thêm nhãn về cảm xúc với 3 class: NEG (tiêu cực), POS (tích cực) và NEU (bình thường) với nội dung nhận xét về sản phẩm quần áo, nhóm đã kiểm data về nhận xét các mẫu điện thoại nhưng chưa tìm thấy nên đã chọn data thay thế. Ban đầu nhóm có thử train với hết tất cả 30000 dữ liệu nhưng do RAM của COLAB không đủ, nên nhóm chỉ lấy 5000 data với phân phối 3 loại cảm xúc đều nhau. Sau đó nhóm có train thử và model tốt nhất trên tập test chỉ đạt 55%. Sau đó nhóm Preprocess lại chọn ra các data chỉ thuộc 2 lớp NEG và POS và chọn ra 5000 data. Các bước clean data nhóm thực hiện như sau.

- + **Bước 1:** Tokenize bằng tool của underthesea
- + **Bước 2:** Loại bỏ các từ stopwords như ừ, à, đó, đây, đến,...
- + **Bước 3:** Loại bỏ icon, hastag, các kí tự đặc biệt cũng như chuyển các dấu thành dấu chấm cuối cùng là viết thường
- + **Bước 4:** ghép các word lại thành 1 câu cũng như bỏ dấu trên đầu thông qua module của underthesea, các từ ghép chẳng hạn: "hàng không" sẽ trở thành "hang_khong".

3.1.2 PhoBert

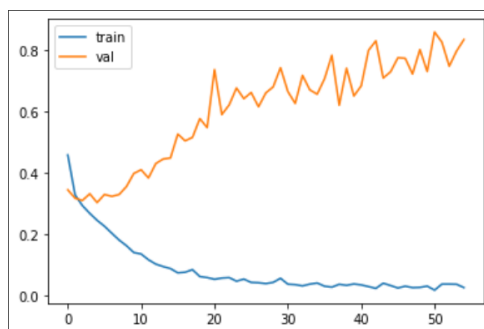
PhoBert là một pretrain model sử dụng mô hình attention network sử dụng cho các tác vụ liên quan đến tiếng Việt. Nó là một module của thư viện transformer hỗ trợ về việc trích xuất các văn bản tiếng Việt thành một lớp embedding với 256 feature. Trong bài tập lớn này, nhóm sử dụng data đã clean ở trên để đưa về một ma trận embedding gồm $5000 * 256$, để từ đó đưa ma trận này vào các mô hình liệt kê bên dưới và kiểm chứng xem model nào là tốt nhất để xử lý văn bản tiếng Việt. Nhóm chọn một mô hình học máy SVM và 4 model Neural Network, bảng kết quả có thể hiện bên dưới. Về việc trích xuất ra các vector đặc trưng, tốn khoảng 1h10 phút cho 5000 data.

3.1.3 SVM

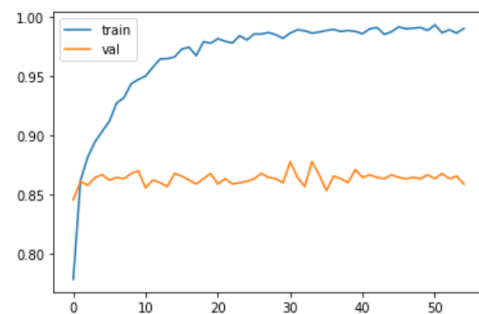
Đây là mô hình ML về việc với một vector mới, sẽ chọn các vector có khoảng cách gần nhất và dựa vào đó để rút trích đặc trưng đưa ra kết quả dự đoán.

3.1.4 LSTM

Đây là mô hình chuỗi sử dụng cho các data dạng seq2seq, việc học đặc trưng mang tính chất toàn cục



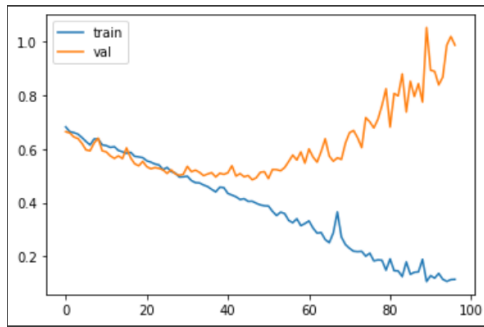
(a) loss



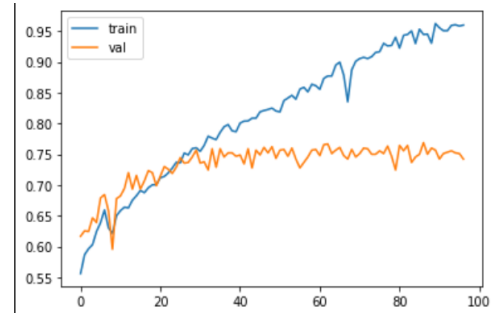
(b) accuracy

3.1.5 biLSTM

Là mô hình cải tiến của LSTM với việc dùng các node ở tương lai để điều chỉnh sai lệch các tính toán trong quá khứ.



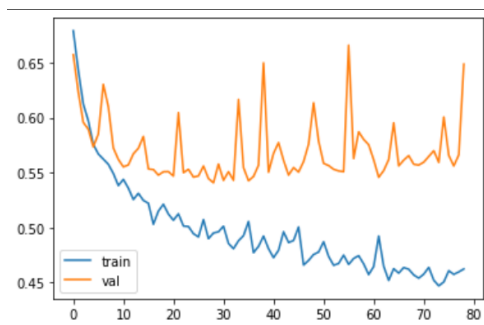
(a) *loss*



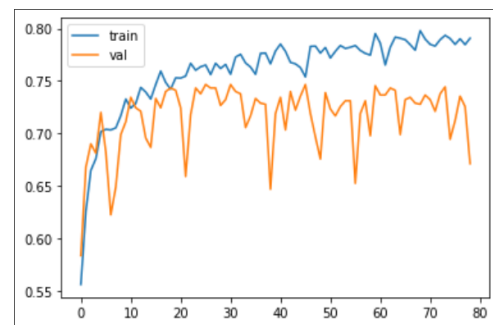
(b) *accuracy*

3.1.6 CNN

Đây là mô hình thường sử dụng trong các tác vụ liên quan đến việc xử lý hình ảnh. Nhưng vẫn có thể sử dụng cho các bài toán văn bản với việc học đặc trưng dựa vào tính cục bộ,



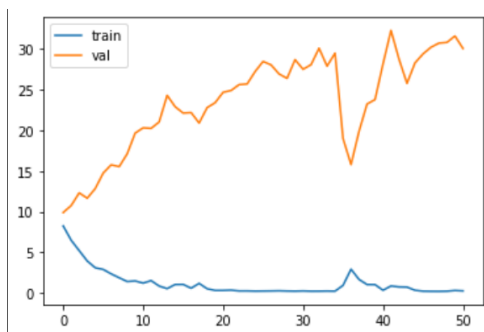
(a) *loss*



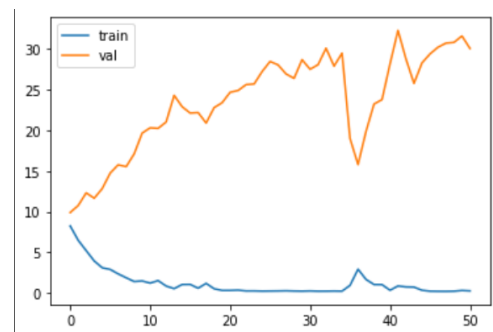
(b) *accuracy*

3.1.7 MLP

Là một mô hình neural network với việc chồng các layer lên nhau, mỗi layer sẽ liên quan tới việc trích xuất đặc trưng khác nhau.



(a) *loss*



(b) *accuracy*

3.1.8 Kết luận về độ chính xác

Do model MLP có độ chính xác cao nhất cũng như độ mất mát thấp nhất nên ta sử dụng model này để dự đoán các dữ liệu Tiếng Việt từ các trang web.

MODEL	loss	accuracy
SVM	NULL	0.876
LSTM	0.3021	0.8740
CNN	0.5620	0.7040
MLP	0.0936	0.8900
biLSTM	0.5053	0.7480

3.2 Dự đoán dữ liệu của Shopee sử dụng MLP

1. Với dữ liệu test:

```
[58] sentence = "áo này đẹp ghê"
predict(sentence)

100%|██████████| 1/1 [00:00<00:00, 1.04it/s]

[0.23089075 0.7691092 ]

'positive'
```

(a) *testcase1*

```
[59] sentence = "cổ tay ko vừa"
predict(sentence)

100%|██████████| 1/1 [00:01<00:00, 1.65s/it]

[0.906933 0.09306694]

'negative'
```

(b) *testcase2*

```
sentence = "dịch vụ khá tốt, tôi rất hài lòng"
predict(sentence)

100%|██████████| 1/1 [00:01<00:00, 1.55s/it]

[0.14915468 0.8508453 ]

'positive'
```

(c) *testcase3*

```
▶ sentence = "mặt hàng còn xấu, đề nghị chỉnh sửa"
predict(sentence)

▶ 100%|██████████| 1/1 [00:01<00:00, 1.39s/it]

[0.6910521 0.30894786]

'negative'
```

(d) *testcase4*

2. Với dữ liệu crawl được từ shopee về mặt hàng áo

```
[58] sentence = "áo này đẹp ghê"
predict(sentence)

100%|██████████| 1/1 [00:00<00:00, 1.04it/s]

[0.23089075 0.7691092 ]

'positive'
```

(a) *quá trình dự đoán*

```
[59] sentence = "cổ tay ko vừa"
predict(sentence)

100%|██████████| 1/1 [00:01<00:00, 1.65s/it]

[0.906933 0.09306694]

'negative'
```

(b) *kết quả*



4 Tài liệu tham khảo

- [MongoDB introduction](#)
- [MongoDB Atlas Introduction](#)
- [BS4 Introduction](#)
- [NaiveBayes Model](#)
- [Colab Model - link my team pretrained](#)
- [Streamlit tutorial](#)